04 EXPLOITING SOFTWARE DIVERSIFICATION FOR WEBASSEMBLY

4.1 Offensive Software Diversification

4.1.2 **Use case 1:** Automatic testing and fuzzing of WebAssembly consumers

TODO We explain the CVE. Make the explanation around "indirect memory diversification"

4.1.3 **Use case 2:** WebAssembly malware evasion

TODO The malware evasion paper

4.2 Defensive Software Diversification

4.2.2 **Use case 3:** Multivariant execution at the Edge

TODO Disturbing of execution time. Go around the web timing attacks.

4.2.3 **Use case 4:** Speculative Side-channel protection

In concrete, distributing the unmodified binary to 100 machines would, essentially, creates 100 homogeneously vulnerable machines. However, let us illustrate the case with a different approach: each time the binary is replicated onto a different machine, we distribute a unique variant instead of the original binary. If we disseminate a unique variant, with X stacked transformations, to each machine, every system would run a distinct Wasm binary. Based on our findings, even when some binaries are still vulnerable, we can confidently say that if 100 variants of a vulnerable program, each furnished with X stacked transformations, are distributed, the impact of any potential attack is considerably mitigated. While it's true that some variants may retain their original vulnerabilities, not all of them do. This significantly enhances overall security. Further reinforcing

this point, let's consider the case of btb_leakage. In this scenario, a suite of 100 variants, each featuring at least 200 stacked transformations, ensures full protection against potential threats, effectively securing the entire infrastructure. Moreover, considering the results for the ret2spec attack, this property holds for the whole population of generated variants, despite the number of stacked transformations. Therefore, WASM-MUTATE as a software diversification tool, is a peemptive solution to potential attacks.

TODO Go around the last paper